

SUPPLEMENT TO “REVERSIBLE MCMC ON MARKOV EQUIVALENCE CLASSES OF SPARSE DIRECTED ACYCLIC GRAPHS”*

BY YANGBO HE[†], JINZHU JIA^{†,‡} AND BIN YU[‡]

*School of Mathematical Sciences and Center of Statistical Science
Peking University[†]*

University of California, Berkeley[‡]

This supplementary material includes three parts: some preliminary results, four examples, an experiment, three new algorithms, and all proofs of the results in the paper [5].

CONTENTS

1	Preliminary results	1
1.1	Characterizations of completed PDAGs	2
1.2	The necessary and sufficient validity conditions	2
1.3	Two algorithms to implement Chickering’s approach	3
2	Additional examples, experiment and algorithms	3
2.1	Examples	3
2.2	Experiment about v-structures	6
2.3	Three Algorithms to check \mathbf{iu}_3 , \mathbf{id}_3 and \mathbf{dd}_2 in Algorithm 1.1	7
3	Proofs	9
3.1	Proof of Theorem 1 in the paper [5]	9
3.2	Proof of Theorem 2 introduced in Subsection 2.3	22
	References	28
	Author’s addresses	28

1. Preliminary results. In this Section, we provide some preliminary results and algorithms introduced by Andersson [1], Dor and Tarsi [4], and Chickering [2, 3]. These results are necessary to implement our proposed approach technically. Some definitions and notation are introduced first. A graph is called a *chain graph* if it contains no partially directed cycles [6].

*This work was supported partially by NSFC (11101008, 11101005, 71271211), 973 Program-2007CB814905, DPHEC-20110001120113, US NSF grants DMS-0907632, CCF-0939370, DMS-0605165, DMS-1107000, SES-0835531 (CDI), ARO grant W911NF-11-1-0114, US NSF Science and Technology Center, LMAM, and LMEQF.

Keywords and phrases: Sparse graphical model, Reversible Markov chain, Markov equivalence class

A *chord* of a cycle is an edge that joins two nonadjacent vertices in the cycle. An undirected graph is *chordal* if every cycle of length greater than or equal to 4 possesses a chord. A directed edge of a DAG is *compelled* if it occurs in the corresponding completed PDAG, otherwise, the directed edge is *reversible* and the corresponding parents are reversible parents. Recall N_x be the set of all neighbors of x , Π_x is the set of all parent of x , $N_{xy} = N_x \cap N_y$ and $\Omega_{x,y} = \Pi_x \cap N_y$ and the concept of “strongly protected” is presented in Definition 1 in the paper [5].

1.1. *Characterizations of completed PDAGs.* Lemma 2 characterizes completed PDAGs that are used to represent Markov equivalence classes [1] and will be used in the proofs in Section 3.

LEMMA 2 (Andersson [1]). *A graph \mathcal{C} is a completed PDAG of a directed acyclic graph \mathcal{D} if and only if \mathcal{C} satisfies the following properties:*

- (i) \mathcal{C} is a chain graph;
- (ii) Let \mathcal{C}_τ be the subgraph induced by τ . \mathcal{C}_τ is chordal for every chain component τ ;
- (iii) $w \rightarrow u - v$ does not occur as an induced subgraph of \mathcal{C} ;
- (iv) Every arrow $v \rightarrow u$ in \mathcal{C} is strongly protected.

1.2. *The necessary and sufficient validity conditions.* Lemma 3 shows the equivalent validity conditions for **iu**₂, **du**₁, **id**₂, **dd**₁ and **mv**₁ used in Definition 9 in the paper [5].

LEMMA 3 (Validity conditions of some operators [3]). *The necessary and sufficient validity conditions of the operators with type of InsertU, DeleteU, InsertD, DeleteD or MakeV are as follows.*

- (InsertU) Let x and y be two vertices that are not adjacent in \mathcal{C} . The operator InsertU $x - y$ is valid (equivalently, **iu**₂ holds) if and only if (iu_{2.1}) $\Pi_x = \Pi_y$, (iu_{2.2}) every undirected path from x to y contains a vertex in N_{xy} .
- (DeleteU) Let $x - y$ be an undirected edge in completed PDAG \mathcal{C} . The operator DeleteU $x - y$ is valid (equivalently, **du**₁ holds) if and only if (du_{1.1}) N_{xy} is a clique in \mathcal{C} .
- (InsertD) Let x and y be two vertices that are not adjacent in \mathcal{C} . The operator InsertD $x \rightarrow y$ is valid (equivalently, **id**₂ holds) if and only if (id_{2.1}) $\Pi_x \neq \Pi_y$, (id_{2.2}) $\Omega_{x,y}$ is a clique, (id_{2.3}) every partially directed path from y to x contains at least one vertex in $\Omega_{x,y}$.

- (*DeleteD*) Let $x \rightarrow y$ be a directed edge in completed PDAG \mathcal{C} . The operator *DeleteD* of $x \rightarrow y$ is valid (equivalently, \mathbf{dd}_1 holds) if and only if $(\mathbf{dd}_{1.1})$ N_y is a clique.
- (*MakeV*) Let $x - z - y$ be any length-two undirected path in \mathcal{C} such that x and y are not adjacent. The operator *MakeV* $x \rightarrow z \leftarrow y$ is valid (equivalently, \mathbf{mv}_1 holds) if and only if $(\mathbf{mv}_{1.1})$ every undirected path between x and y contains a vertex in N_{xy} .

1.3. *Two algorithms to implement Chickering's approach.* Algorithm 3 generates a consistent extension of a PDAG [4]. Algorithm 4 creates the corresponding completed PDAG of a DAG [2]. They are used to implement Chickering's approach.

Algorithm 3: (Dor and Tarsi [4]) Generate a consistent extension of a PDAG

Input: A PDAG \mathcal{P} that admits a consistent extension

Output: A DAG \mathcal{D} that is a consistent extension of \mathcal{P} .

```

1 Let  $\mathcal{D} := \mathcal{P}$ ;
2 while  $\mathcal{P}$  is not empty do
3     Select a vertex  $x$  in  $\mathcal{P}$  such that (1)  $x$  has no outgoing edges and (2) if  $N_x$  is not
       empty, then every vertex in  $N_x$  is adjacent to all vertices in  $N_x \cup \Pi_x$ . /* Dor
       and Tarsi [4] show that a vertex  $x$  with these properties is
       guaranteed to exist if  $\mathcal{P}$  admits a consistent extension. */
4     Let all undirected edges adjacent to  $x$  be directed toward  $x$  in  $\mathcal{D}$ 
5     Remove  $x$  and all incident edges from  $\mathcal{P}$ .
6 return  $\mathcal{D}$ 
```

2. Additional examples, experiment and algorithms. This section include three parts: (1) some examples to illuminate the methods proposed in the paper [5], (2) an experiment about v-structures, and (3) three algorithms to test the conditions \mathbf{iu}_3 , \mathbf{id}_3 and \mathbf{dd}_2 in Algorithm 1.1 only based on e_t in an efficient manner.

2.1. *Examples.* Four examples are presented to illustrate operators, the generation of a resulting completed PDAG of an operator, the conditions of a perfect operator set, and the process of constructing a perfect operator set.

Example 1. This example illustrates six operators on a completed PDAG \mathcal{C} and their corresponding modified graphs. Figure 8 displays six operators: *InsertU* $x - z$, *DeleteU* $y - z$, *InsertD* $x \rightarrow v$, *DeleteD* $z \rightarrow v$, *MakeV* $z \rightarrow y \leftarrow u$, and *RemoveV* $z \rightarrow v \leftarrow u$. After inserting an undirected edge

Algorithm 4: (Chickering [2]) Create the completed PDAG of a DAG

Input: \mathcal{D} , a DAG

Output: The completed PDAG \mathcal{C} of DAG \mathcal{D} .

```

1 Perform a topological sort on the vertices in  $\mathcal{D}$  such that for any pair of vertices  $x$ 
  and  $y$  in  $\mathcal{D}$ ,  $x$  must precede  $y$  if  $x$  is an ancestor of  $y$ ;
2 Sort the edges first in ascending order for incident vertices and then in descending
  order for outgoing vertices; Label every edge in  $\mathcal{D}$  as unknown;
3 while there are edges labeled unknown in  $\mathcal{D}$  do
4   Let  $x \rightarrow y$  be the lowest ordered edge that is labeled unknown
5   for every edge  $w \rightarrow x$  labeled compelled do
6     if  $w$  is not a parent of  $y$  then
7        $x \rightarrow y$  and every edge incident into  $y$  with compelled
8       Goto 3
9     else
10      Label  $w \rightarrow y$  with compelled
11   if there exists an edge  $z \rightarrow y$  such that  $z = x$  and  $z$  is not a parent of  $x$  then
12     Label  $x \rightarrow y$  and all unknown edges incident into  $y$  with compelled
13   else
14     Label  $x \rightarrow y$  and all unknown edges incident into  $y$  with reversible
15 Let  $\mathcal{C} = \mathcal{D}$  and undirect all edges labeled "reversible" in  $\mathcal{C}$ .
16 return completed PDAG  $\mathcal{C}$ 

```

$x - z$ into the initial graph \mathcal{C} , we get a modified graph denoted as \mathcal{P}_1 in Figure 8. By applying the other five operators to \mathcal{C} in Figure 8 respectively, we can obtain other five corresponding modified graphs $\mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5$, and \mathcal{P}_6 . Here the operator "MakeV $z \rightarrow y \leftarrow u$ " modifies $z - y - u$ to $z \rightarrow y \leftarrow u$ and the operator "Remove $z \rightarrow v \leftarrow u$ " modifies $z \rightarrow v \leftarrow u$ to $z - v - u$. Notice that a modified graph might not be a PDAG though all modified graphs in this example are PDAGs.

In the above example, we see that the modified graph of an operator, denoted by \mathcal{P} , might be a PDAG, but might not be a completed PDAG. For example, the modified graphs \mathcal{P}_4 , and \mathcal{P}_6 in Figure 8 are not completed PDAGs because the directed edge $y \rightarrow v$ is not strongly protected.

Example 2. This example illustrates Chickering's approach to obtain the resulting completed PDAG of a valid operator from its modified graph. Consider the initial completed PDAG \mathcal{C} and the operator "Remove $z \rightarrow v \leftarrow u$ " in Figure 8. We illustrate in Figure 9 the steps of Chickering's approach that generates the resulting completed PDAG \mathcal{C}_1 by applying "Remove $z \rightarrow v \leftarrow u$ " to \mathcal{C} . The first step (step 1) extends the modified graph (a PDAG \mathcal{P}_6) to a consistent extension (\mathcal{D}_6) via Algorithm 3. The second step (step

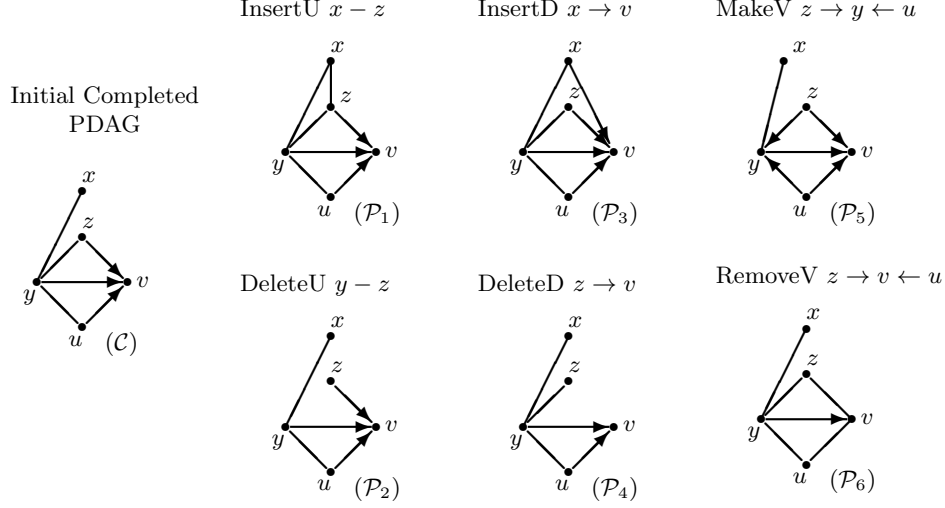


FIG 8. Examples of six operators of PDAG \mathcal{C} . \mathcal{P}_1 to \mathcal{P}_6 are the modified graphs of six operators.

2) constructs the resulting completed PDAG \mathcal{C}_1 of the operator “Remove $z \rightarrow v \leftarrow u$ ” from the DAG \mathcal{D}_6 via Algorithm 4.

Example 3. This example illustrates that \mathcal{O} in Equation (3.3) will not be reversible if condition **iu**₃ or **dd**₂ is not contained in Definition 9. Consider the operator set \mathcal{O} defined in Equation (3.3) for \mathcal{S}_5 and the completed PDAG $\mathcal{C} \in \mathcal{S}_5$ in Figure 10. We have that operator InsertU $z - u$ and DeletedD $z \rightarrow v$ are valid. As shown in Figure 10, InsertU $z - u$ transfers \mathcal{C} to the completed PDAG \mathcal{C}_1 and DeletedD $z \rightarrow v$ transfers \mathcal{C} to the completed PDAG \mathcal{C}_2 . However, deleting $z - u$ from \mathcal{C}_1 will result in an undirected PDAG distinct from \mathcal{C} and InsertD $z \rightarrow v$ is not valid for \mathcal{C}_2 . As a consequence, if \mathcal{O} contains InsertU $z - u$ and DeletedD $z \rightarrow v$, it will be not reversible. According to Definition 9, these two operators do not appear in $\mathcal{O}_{\mathcal{C}}$ because they do not satisfy the conditions **iu**₃ and **dd**₂ respectively.

Example 4. This toy example is given to show how to construct a concrete perfect set of operators following Definition 9 in the paper [5]. Consider the completed PDAG \mathcal{C} in Example 3. Here we introduce the procedure to determine $\text{InsertU}_{\mathcal{C}}$. All possible operators of inserting an undirected edge to \mathcal{C} include: “InsertU $x - z$ ”, “InsertU $x - u$ ”, “InsertU $x - v$ ” and “InsertU $z - u$ ”. The operator “InsertU $x - v$ ” is not valid according to Lemma 3

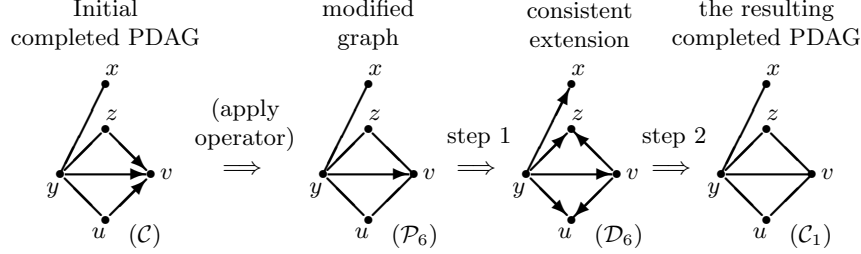


FIG 9. Example for constructing the unique resulting completed PDAG of a valid operator. An operator “Remove $z \rightarrow v \leftarrow u$ ” in Figure 8 is applied to the initial completed PDAG \mathcal{C} and finally results in the resulting completed PDAG \mathcal{C}_1 .

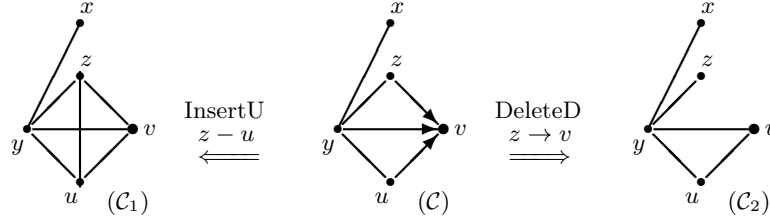


FIG 10. Example: Two valid operators bring about irreversibility. It shows valid conditions are not sufficient for perfect operator set.

since $\Pi(x) \neq \Pi(v)$. The operator “InsertU $z - u$ ” is valid; however, condition **iu**₃ does not hold. According to Definition 9 in the paper [5], we have that only “InsertU $x - z$ ” and “InsertU $x - u$ ” are in $InsertU_{\mathcal{C}}$. Thus $InsertU_{\mathcal{C}} = \{x - z, x - u\}$, where “ $x - z$ ” denotes “InsertU $x - z$ ” in the set. Table 1 lists the six sets of operators on \mathcal{C} .

2.2. Experiment about v -structures. Below, we present the experiment result in Figure 11 about the numbers of v -structures of completed PDAGs in \mathcal{S}_p^{rp} .

For $\mathcal{S}_p^{1.5p}$ in the main window, the medians of the four distributions are 108, 220, 557 and 1110 for $p=100, 200, 500$, and 1000 respectively. Figure 11 shows that the numbers of v -structures are much less than (p^2) for most completed PDAGs in \mathcal{S}_p^{rp} when r is set to 1.2, 1.5 or 3. This result is useful to

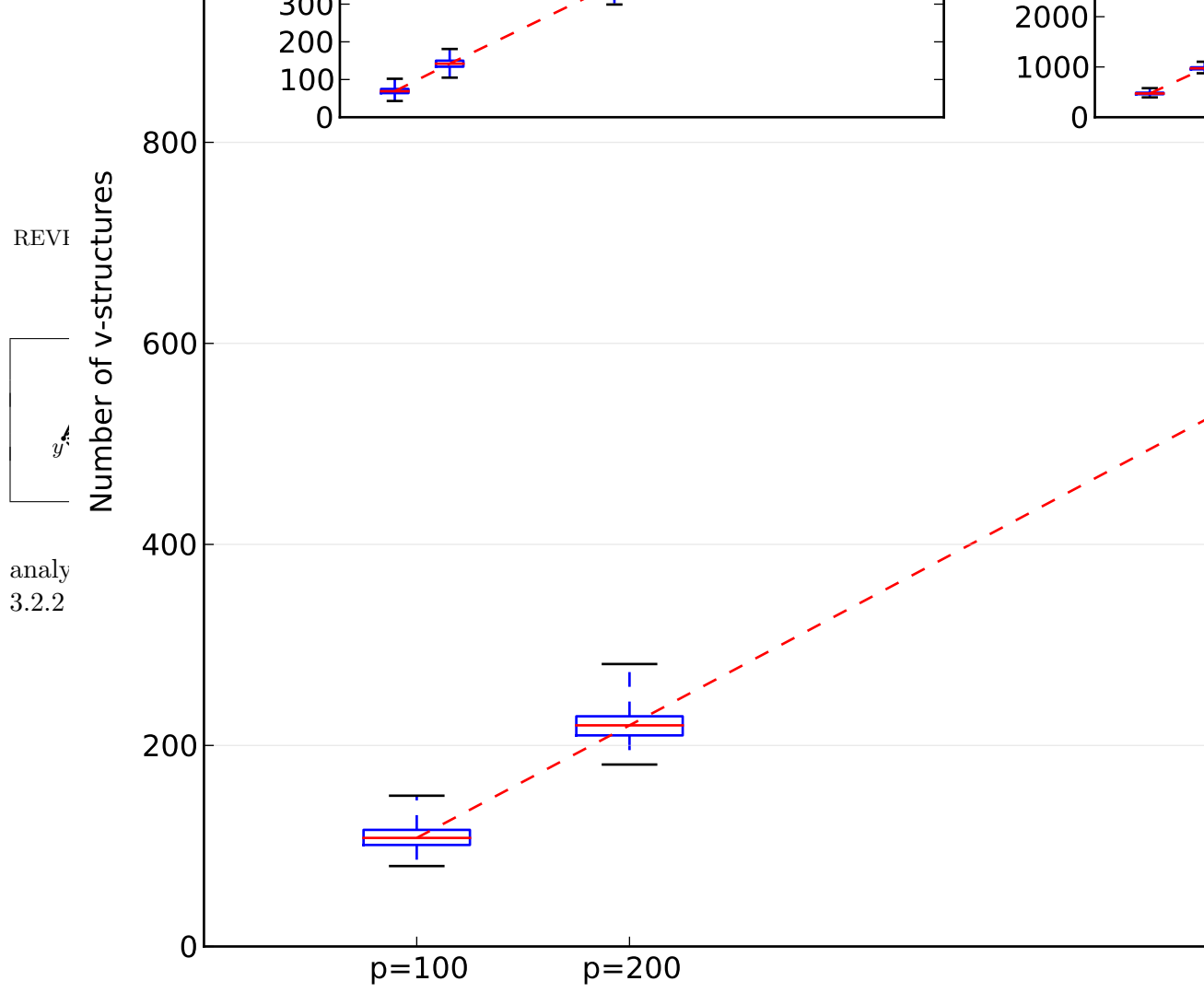


FIG 11. The distributions of the numbers of v -structures of completed PDAGs in S_p^{rp} . The red lines in the boxes indicate the medians.

2.3. *Three Algorithms to check \mathbf{iu}_3 , \mathbf{id}_3 and \mathbf{dd}_2 in Algorithm 1.1.* The conditions \mathbf{iu}_3 , \mathbf{id}_3 and \mathbf{dd}_2 in the fourth group depend on both e_t and the resulting completed PDAGs of the operators. Intuitively, checking these three conditions requires that we obtain the corresponding resulting completed PDAGs. We know that the time complexity of getting a resulting completed PDAG of e_t is $O(pn_{e_t})$ [3, 4], where n_{e_t} is the number of edges in e_t . To avoid generating resulting completed PDAG, we provide three algorithms to check \mathbf{iu}_3 , \mathbf{id}_3 and \mathbf{dd}_2 respectively.

In these three algorithms, we use the concept of strongly protected edges, defined in Definition 2. Let Δ_v contain all vertices adjacent to v . To check whether a directed edge $v \rightarrow u$ is strongly protected or not in a graph \mathcal{G} , from Definition 2, we need to check whether one of the four configurations in Figure 1 occurs in \mathcal{G} . This can be implemented by local search in Δ_v and

Δ_u . We know that when a PDAG is sparse, in general, these sets are small, so it is very efficient to check whether an edge is “strongly protected”.

We are now ready to provide Algorithm 1.1.1, Algorithm 1.1.2, and Algorithm 1.1.3 to check \mathbf{iu}_3 , \mathbf{id}_3 and \mathbf{dd}_2 only based on e_t , respectively. In these three algorithms, we just need to check whether a few directed edges are strongly protected or not in \mathcal{P}_{t+1} , which has only one or a few edges different from e_t . We prove in Theorem 2 that these three algorithms are equivalent to checking conditions \mathbf{iu}_3 , \mathbf{id}_3 and \mathbf{dd}_2 , respectively.

Algorithm 1.1.1: Check the condition \mathbf{iu}_3 in Definition 9

Input: a completed PDAG e_t and a valid operator on it: InsertU $x - y$.

Output: True or False

```

1 Insert  $x - y$  to  $e_t$ , get the modified PDAG denoted as  $\mathcal{P}_{t+1}$ ,
2 for each common child  $u$  of  $x$  and  $y$  in  $\mathcal{P}_{t+1}$  do
3   | if either  $x \rightarrow u$  or  $y \rightarrow u$  is not strongly protected in  $\mathcal{P}_{t+1}$  then
4   |   | return False
5 return True ( $\mathbf{iu}_3$  holds for InsertU  $x - y$ )

```

Algorithm 1.1.2: Check the condition \mathbf{id}_3 in Definition 9

Input: a completed PDAG e_t and a valid operator: InsertD $x \rightarrow y$.

Output: True or False

```

1 Insert  $x \rightarrow y$  to  $e_t$ , get a PDAG, denoted as  $\mathcal{P}_o$ ,
2 for each undirected edge  $u - y$  in  $\mathcal{P}_o$ , where  $u$  is not adjacent to  $x$  do
3   | update  $\mathcal{P}_o$  by orienting  $u - y$  to  $y \rightarrow u$ ,
4 for each edge  $v \rightarrow y$  in  $\mathcal{P}_o$  do
5   | if  $v \rightarrow y$  is not strongly protected in  $\mathcal{P}_o$  then
6   |   | update  $\mathcal{P}_o$  by changing  $v \rightarrow y$  to  $v - y$ ,
7 Set  $\mathcal{P}_{t+1} = \mathcal{P}_o$ 
8 for each common child  $u$  of  $x$  and  $y$  in  $\mathcal{P}_{t+1}$  do
9   | if  $y \rightarrow u$  is not strongly protected in  $\mathcal{P}_{t+1}$  then
10  |   | return False
11 return True ( $\mathbf{id}_3$  holds for InsertD  $x \rightarrow y$ )

```

THEOREM 2 (Correctness of Algorithms 1.1.1, 1.1.2 and 1.1.3). *Let e_t be a completed PDAG. We have the following results.*

- (i) *Let InsertU $x - y$ be any valid operator of e_t , then condition \mathbf{iu}_3 holds for the operator InsertU $x - y$ if and only if the output of Algorithm*

Algorithm 1.1.3: Check the condition \mathbf{dd}_2 in Definition 9

Input: a completed PDAG e_t and a valid operator $\text{DeleteD } x \rightarrow y$

Output: True or False

```

1 Delete  $x \rightarrow y$  from  $e_t$ , get a PDAG, denoted as  $\mathcal{P}_{t+1}$ ;
2 for each parent  $v$  of  $y$  in  $\mathcal{P}_{t+1}$  do
3   if  $v \rightarrow y$  is not strongly protected in  $\mathcal{P}_{t+1}$  then
4     return False
5 return True ( $\mathbf{dd}_2$  holds for  $\text{DeleteD } x \rightarrow y$ )
    
```

1.1.1 is True.

- (ii) Let $\text{InsertD } x \rightarrow y$ be any valid operator of e_t , then condition \mathbf{id}_3 holds for the operator $\text{InsertD } x \rightarrow y$ if and only if the output of Algorithm 1.1.2 is True.
- (iii) Let $\text{DeleteD } x \rightarrow y$ be any valid operator of e_t , then condition \mathbf{dd}_2 holds for the operator $\text{DeleteD } x \rightarrow y$ if and only if the output of Algorithm 1.1.3 is True.

In Theorem 2, we show that an algorithm (Algorithm 1.1.1, Algorithm 1.1.2, or Algorithm 1.1.3) returns True for an operator if and only if the corresponding condition (\mathbf{iu}_3 , \mathbf{id}_3 or \mathbf{dd}_2) holds for the operator. Theorem 2 says that we do not have to examine the resulting completed PDAG to check conditions \mathbf{iu}_3 , \mathbf{id}_3 and \mathbf{dd}_2 , which saves much computation time.

3. Proofs. We will provide a proof of Theorem 1, the main result of the paper [5], in Subsection 3.1, and a proof of Theorem 2 in Subsection 3.2 below. Notice that we present Theorem 2 in Subsection 2.3 to show the correctness of Algorithm 1.1.1, Algorithm 1.1.2 and Algorithm 1.1.3.

3.1. *Proof of Theorem 1 in the paper [5].* Let \mathcal{O} be the operator set defined in Equation (3.3); to prove Theorem 1, which shows \mathcal{O} is a perfect operator set, we need to show \mathcal{O} satisfies four properties: validity, distinguishability, irreducibility and reversibility. Equivalently, we just need to prove Theorem 3–6 as follows:

THEOREM 3. *The operator set \mathcal{O} is valid.*

THEOREM 4. *The operator set \mathcal{O} is distinguishable.*

THEOREM 5. *The operator set \mathcal{O} is reversible.*

THEOREM 6. *The operator set \mathcal{O} is irreducible.*

Of the above four theorems, the most important and difficult is to prove Theorem 5. We now show the proofs one by one.

Proof of Theorem 3

According to the definition of validity in Definition 5 and the definition of \mathcal{O}_C in Equation (3.2), all operators in $InsertU_C$, $DeleteU_C$, $InsertD_C$, $DeleteD_C$ and $MakeV_C$ are valid. We just need to prove Lemma 4, which shows all operators in $RemoveV_C$ are valid.

LEMMA 4. *Let $x \rightarrow z \leftarrow y$ be a v-structure in completed PDAG \mathcal{C} . If $(\mathbf{rv}_1) \Pi_x = \Pi_y$, $(\mathbf{rv}_2) \Pi_x \cup N_{xy} = \Pi_z \setminus \{x, y\}$, and (\mathbf{rv}_3) every undirected path between x and y contains a vertex in N_{xy} hold, then the operator $RemoveV$ $x \rightarrow z \leftarrow y$ is valid and results in a completed PDAG in \mathcal{S}_p^n defined in Equation (3.1).*

To prove Lemma 4, we will use Lemma 5 given by Chickering (Lemma 32 in [3]).

LEMMA 5. *Let \mathcal{C} be any completed PDAG, and let x and y be any pair of vertices that are not adjacent. Every undirected path between x and y passes through a vertex in N_{xy} if and only if there exists a consistent extension in which (1) x has no reversible parents, (2) all vertices in N_{xy} are parents of y , and (3) y has no other reversible parents.*

We now give a proof of Lemma 4.

PROOF. From Lemma 5 and condition \mathbf{rv}_3 in Lemma 4, there exists a consistent extension of \mathcal{C} , denoted by \mathcal{D} , in which x has no reversible parents and the reversible parents of y are the vertices in N_{xy} . Because $y \rightarrow z$ occurs in the completed PDAG \mathcal{C} , N_z and N_y occur in different chain components. We can orient the undirected edges adjacent to z out of z . Then all vertices in N_z are children of z in \mathcal{D} . Let \mathcal{D}' be the graph obtained by reversing $y \rightarrow z$ in \mathcal{D} and \mathcal{P}' be the PDAG obtained by applying the $RemoveV$ operator to \mathcal{C} . We will show that \mathcal{D}' is a consistent extension of \mathcal{P}' .

Clearly, \mathcal{D}' and \mathcal{P}' have the same skeleton.

We have that any v-structure that occurs in \mathcal{D} but not in \mathcal{P}' must include either the edge $x \rightarrow z$ or $y \rightarrow z$. Since \mathcal{D} is a consistent extension of \mathcal{C} , we have that all v-structures in \mathcal{D} are also in \mathcal{C} . From condition \mathbf{rv}_2 , all parents of z other than x and y are adjacent to x and y . Hence $x \rightarrow z \leftarrow y$ is the only v-structure that is directed into z in \mathcal{C} . We have that all v-structures

of \mathcal{P}' are also in \mathcal{D} , and there is only one v-structure $x \rightarrow z \leftarrow y$ that is in \mathcal{D} but not \mathcal{P}' .

Since $y \rightarrow z$ is the unique edge that differs between \mathcal{D} and \mathcal{D}' , we have that any v-structure that exists in \mathcal{D} but not in \mathcal{D}' must include the edge $y \rightarrow z$, and any v-structure that exists in \mathcal{D}' but not in \mathcal{D} must include the edge $z \rightarrow y$. We have shown that $x \rightarrow z \leftarrow y$ is the only v-structure in \mathcal{D} that is directed into z . From the construction of \mathcal{D} , we have that all compelled parents of y in \mathcal{D}' are also parents of z and all other parents are in N_{xy} ; from rv_2 , they also are parents of z . There is no v-structure that includes edge $z \rightarrow y$ in \mathcal{D}' . Hence, all v-structures of \mathcal{D}' are also in \mathcal{D} , and there is only one v-structure $x \rightarrow z \leftarrow y$ that is in \mathcal{D} but not \mathcal{D}' .

Hence, \mathcal{D}' and \mathcal{P}' have the same v-structures. It remains to be shown that \mathcal{D}' is acyclic.

If \mathcal{D}' contains a cycle, the cycle must contain the edges $z \rightarrow y$ because \mathcal{D} is acyclic. This implies there is a directed path from y to z in \mathcal{D} . By construction, all vertices in N_z are children of z in \mathcal{D}' . So, this path must include a compelled parent of z , denote it by u . If $u \neq x$, from condition rv_2 , $u \in \Pi_y \cup N_{xy}$; by the construction of \mathcal{D} , we have $u \in \Pi_y$. Thus, there is no path from y to z that contains u . If $u = x$, by construction, the path must contain a compelled parent v of x . From condition rv_1 , $v \in \Pi_y$. Thus, there is no path from y to z contains v . We get that \mathcal{D}' is acyclic. Thus \mathcal{D}' is a consistent extension of \mathcal{P}' and the operator $\text{RemoveV } x \rightarrow z \leftarrow y$ is valid. \square

Proof of Theorem 4

PROOF. For any completed $\mathcal{C} \in \mathcal{S}_p^n$, we need to show that different operators in $O_{\mathcal{C}}$ result in different completed PDAGs. For any valid operator $o \in \text{InsertU}_{\mathcal{C}}$, say $\text{InsertU } x - y$, denoted as o , the resulting completed PDAG of o contains the undirected edge $x - y$. We have that all other operators in $O_{\mathcal{C}}$ except for $\text{InsertD } x \rightarrow y$ and $\text{Insert } x \leftarrow y$ (if they are also valid) will result in completed PDAGs with skeletons different than the resulting completed PDAG of o . Thus, these operators can not result in the same completed PDAG as o . If $\text{InsertD } x \rightarrow y$ or $\text{Insert } x \leftarrow y$ is valid, the resulting completed PDAGs of them contain $x \rightarrow y$ or $x \leftarrow y$. These two resulting completed PDAGs have at least a compelled edge different than the resulting completed PDAG of o . Thus, there is no operator in $O_{\mathcal{C}}$ that can result in the same completed PDAG as o .

Similarly, we can show for any operator in $\mathcal{O}_{\mathcal{C}}$, different operators will result in different completed PDAGs, because they will have distinct skeletons, compelled edges, or v-structures. \square

Proof of Theorem 5.

Let \mathcal{C} be any completed PDAG in \mathcal{S}_p^n , $o \in \mathcal{O}_{\mathcal{C}}$ be an operator on \mathcal{C} . The operator $o' \in \mathcal{O}$ is the *reversible operator* of o if o' can transfer the resulting completed PDAG of o back to \mathcal{C} . To prove Theorem 5, we just need to show each operator in $\mathcal{O}_{\mathcal{C}}$ defined in Equation (3.3) has a reversible operator in \mathcal{O} . Equivalently, we prove Lemma 6, Lemma 7, Lemma 8, Lemma 9, Lemma 10, and Lemma 11 to show the reversibility for six types of operators respectively.

LEMMA 6. *For any operator $o \in \mathcal{O}_{\mathcal{C}}$ denoted by “InsertU $x - y$ ”, the operator “DeleteU $x - y$ ” is the reversible operator of o .*

LEMMA 7. *For any operator $o \in \mathcal{O}_{\mathcal{C}}$ denoted by “DeleteU $x - y$ ”, the operator “InsertU $x - y$ ” is the reversible operator of o .*

LEMMA 8. *For any operator $o \in \mathcal{O}_{\mathcal{C}}$ denoted by “InsertD $x \rightarrow y$ ”, the operator “DeleteD $x \rightarrow y$ ” is the reversible operator of o .*

LEMMA 9. *For any operator $o \in \mathcal{O}_{\mathcal{C}}$ denoted by “DeleteD $x \rightarrow y$ ”, the operator “InsertD $x \rightarrow y$ ” is the reversible operator of o .*

LEMMA 10. *For any operator $o \in \mathcal{O}_{\mathcal{C}}$ denoted by “MakeV $x \rightarrow z \leftarrow y$ ”, the operator “RemoveV $x \rightarrow z \leftarrow y$ ” is the reversible operator of o .*

LEMMA 11. *For any operator $o \in \mathcal{O}_{\mathcal{C}}$ denoted by “RemoveV $x \rightarrow z \leftarrow y$ ”, the operator “MakeV $x \rightarrow z \leftarrow y$ ” is the reversible operator of o .*

Before giving proofs of these six lemmas, We first provide several results shown in Lemma 12, 14, 13, and Lemma 15.

LEMMA 12. *Let graph \mathcal{C} be a completed PDAG, $\{w, v, u\}$ be three vertices that are adjacent each other in \mathcal{C} . If there are two undirected edges in $\{w, v, u\}$, then the third edge is also undirected.*

PROOF. If the third edge is directed, there is a directed cycle like $w - v - u \rightarrow w$. From Lemma 2, we know that \mathcal{C} is a chain graph, so there is no directed circle in \mathcal{C} . \square

LEMMA 13. *Let \mathcal{C}_1 be the resulting completed PDAG obtained by inserting a new edge between x and y in \mathcal{C} . If there is at least one edge $v \rightarrow u$ that is directed in \mathcal{C} but not directed in \mathcal{C}_1 , then there exists a vertex h that is common child of x and y such that $x \rightarrow h$ and $y \rightarrow h$ in \mathcal{C} become undirected in \mathcal{C}_1 .*

PROOF. According to Lemma 2, an edge is directed in a completed PDAG if and only if it is strongly protected. Thus, we have that at least one case among (a), (b), (c), (d) in Figure 1 occurs in \mathcal{C} but not in \mathcal{C}_1 for $v \rightarrow u$. We will show that either Lemma 13 holds or there exists a parent of u , denoted as u_1 , such that $u_2 \rightarrow u_1$ occurs in \mathcal{C} but not in \mathcal{C}_1 , where u_2 is a parent of u_1 . We denote the latter result as (*).

Suppose case (a) in Figure 1 occurs in \mathcal{C} but not in \mathcal{C}_1 . Because $v \rightarrow u$ becomes undirected in \mathcal{C}_1 , we have that $w \rightarrow v$ must be undirected in \mathcal{C}_1 since w and u are not adjacent. Set $u_1 = v$ and $u_2 = u$, and we have that (*) holds.

Suppose case (b) in Figure 1 occurs in \mathcal{C} but not in \mathcal{C}_1 . If the pair $\{v, w\}$ is not $\{x, y\}$, $v \rightarrow u \leftarrow w$ is a v-structure in \mathcal{C} . We have that $v \rightarrow u$ occurs in \mathcal{C}_1 . This is a contradiction. If $\{v, w\}$ is $\{x, y\}$, we have that Lemma 13 holds ($h = u$).

Suppose case (c) in Figure 1 occurs in \mathcal{C} but not in \mathcal{C}_1 . Either $v \rightarrow w$ or $w \rightarrow u$ occurs in \mathcal{C} but not in \mathcal{C}_1 . If it is $v \rightarrow w$, by setting $u_2 = v$ and $u_1 = w$, we have (*) holds. If it is $w \rightarrow u$, both $v - u$ and $w - u$ in \mathcal{C}_1 , so $x - u$ also must be in \mathcal{C}_1 . We also have that (*) holds.

Suppose case (d) in Figure 1 occurs in \mathcal{C} but not in \mathcal{C}_1 . If the pair $\{w, w_1\}$ is $\{x, y\}$, Lemma 13 holds ($h = u$). Otherwise, $w \rightarrow u \leftarrow w_1$ must occur in both \mathcal{C}_1 and \mathcal{C} and the edge $v \rightarrow u$ is still strongly protected in \mathcal{C}_1 , yielding a contradiction.

If (*) holds, we have that there is a directed path $u_2 \rightarrow u_1 \rightarrow u$ such that $u_2 \rightarrow u_1$ occurs in \mathcal{C} but not \mathcal{C}_1 . Iterating, we can get a directed path $u_k \rightarrow u_{k-1} \cdots \rightarrow u$ of length $k - 1$ without undirected edges such that $u_k \rightarrow u_{k-1}$ occurs in \mathcal{C} but not in \mathcal{C}_1 if Lemma 13 does not hold in each step. Because \mathcal{C} is a chain graph without directed circle, the procedure will stop in finite steps and Lemma 13 will hold eventually. \square

From the proof of Lemma 13, we have that u should be a descendant of x and y , so we can get the following Lemma 14.

LEMMA 14. *Let \mathcal{C} be any completed PDAG, and let \mathcal{P} denote the PDAG that results from adding a new edge between x and y . For any edge $v \rightarrow u$ in \mathcal{C} that does not occur in the resulting completed PDAG extended from \mathcal{P} , there is a directed path of length zero or more from both x and y to u in \mathcal{C} .*

LEMMA 15. *Let $\text{Insert}U_{\mathcal{C}}$ and $\text{Delete}U_{\mathcal{C}}$ be the operator sets defined in Definition 9 respectively. For any o in $\text{Insert}U_{\mathcal{C}}$ or in $\text{Delete}U_{\mathcal{C}}$, where \mathcal{P}' is the modified graph of o that is obtained by applying o to \mathcal{C} , we have that \mathcal{P}' is a completed PDAG.*

PROOF. We just need to check whether \mathcal{P}' satisfies the four conditions in Lemma 2.

(i): For any $o \in DeleteU_{\mathcal{C}}$, denoted as Deleted $x-y$, let \mathcal{P}' be the modified graph obtained by deleting $x-y$ from \mathcal{C} .

If there is a directed cycle in \mathcal{P}' , it must be a directed cycle in \mathcal{C} , which is a contradiction. Thus, there is no directed cycle in \mathcal{P}' and \mathcal{P}' is a chain graph.

If there exists an undirected cycle of length greater than 3 without a chord in \mathcal{P}' , the cycle must contain both x and y ; otherwise, this cycle occurs in \mathcal{C} . If the length of the cycle is 4, the other two vertices are in N_{xy} ; we have that the cycle has a chord since N_{xy} is a clique in \mathcal{C} . If the cycle in \mathcal{P}' has length greater than 4 without a chord, we have that $x-y$ is the unique chord of this cycle in \mathcal{C} . However, this would imply that there is a cycle of length greater than 3 without a chord in \mathcal{C} , a contradiction. Thus, there is no undirected cycle with length greater than 3 in \mathcal{P}' , so every chain component of \mathcal{P}' is chordal.

Suppose that $\cdot \rightarrow \cdot - \cdot$ occurs as an induced subgraph of \mathcal{P}' ; it must be $x \rightarrow \cdot - y$ (or $y \rightarrow \cdot - x$). However, in this case, $x \rightarrow \cdot - y - x$ (or $y \rightarrow \cdot - x - y$) would be a directed cycle in \mathcal{C} . Thus the induced subgraph like $\cdot \rightarrow \cdot - \cdot$ does not occur as an induced subgraph of \mathcal{P}' .

Finally, all directed edges in \mathcal{P}' will be strongly protected; by the definition of strong protection, all directed edges in \mathcal{C} will remain strongly protected when an undirected edge is removed.

(ii): For any $o \in InsertU_{\mathcal{C}}$, denoted as InsertU $x-y$, \mathcal{P}' is the modified graph of o .

If there is a directed cycle in \mathcal{P}' , it must contain $x-y$, otherwise this cycle is also in \mathcal{C} . We can suppose that there exists a partially directed path from x to y in \mathcal{C} . Denote the adjacent vertex of y in the path as u . Let u be the vertex adjacent to y in the path. We have $u \notin \Pi_y$; otherwise, from the condition $\Pi_x = \Pi_y$ in Lemma 3, u would also be in Π_x , so there would be a partially directed cycle from x to x in \mathcal{C} . Hence the directed path must have the form $x \cdots \rightarrow \cdots u - y$. This would induce a subgraph like $a \rightarrow b - v$ in \mathcal{C} , a contradiction. Consequently, \mathcal{P}' is a chain graph.

If there exists an undirected cycle of length greater than 3 without a chord in \mathcal{P}' , the cycle must contain x and y , and there must be an undirected path from x to y in \mathcal{C} ; otherwise, the cycle would also be in \mathcal{C} . From Lemma 3, every undirected path from x to y contains a vertex in N_{xy} , so every undirected path of length greater than two has a chord. Thus, every undirected path of length greater than 3 from x to y in \mathcal{P}' has a chord. This implies

that every chain component of \mathcal{P}' is chordal.

Suppose that a subgraph like $\cdot \rightarrow \cdot - \cdot$ occurs as an induced subgraph of \mathcal{P}' . Since $\Pi_x = \Pi_y$ in \mathcal{C} , the induced subgraph is not $\cdot \rightarrow x - y$ (or $\cdot \rightarrow y - x$). Thus, the induced subgraph like $\cdot \rightarrow \cdot - \cdot$ also occurs in \mathcal{C} . This is a contradict since \mathcal{C} is a completed PDAG, yielding a contradiction.

From Lemma 13 and the condition **iu**₃ in Definition 9, all directed edges in \mathcal{C} are also directed in \mathcal{C}_1 . This implies that all directed edges in \mathcal{P} are still compelled, and are thus strongly protected. \square

We now give proofs of Lemma 6, Lemma 7, Lemma 8, Lemma 9, Lemma 10, and Lemma 11 one by one.

Proof of Lemma 6

PROOF. Because the operator “InsertU $x - y$ ” = $o \in \mathcal{O}_{\mathcal{C}}$ is valid and \mathcal{C}_1 is the resulting completed PDAG of o , we have that $x - y$ occurs in \mathcal{C}_1 . We just need to show that the common neighbors of x and y , denoted as N_{xy} , form a clique in \mathcal{C}_1 .

If N_{xy} is empty set or has only one vertex, the condition that N_{xy} is a clique in \mathcal{C}_1 holds.

If there are two different vertices $z, u \in N_{xy}$ in \mathcal{C}_1 , we have that $x - z - y$ and $x - u - y$ form a cycle of length of 4 in \mathcal{C}_1 . The cycle is also in \mathcal{C} . Since the edge $x - y$ does not exist in \mathcal{C} and \mathcal{C} is a completed PDAG in which all undirected subgraphs are chordal graphs, we have that $z - u$ occurs in \mathcal{C} , so z and u are adjacent in \mathcal{C}_1 . Hence the condition that N_{xy} is a clique in \mathcal{C}_1 holds. \square

Proof of Lemma 7

PROOF. We need to show the operator $o' := \text{InsertU } x - y$, satisfies the conditions **iu**₁, **iu**₂ and **iu**₃ in Definition 9 for completed PDAG \mathcal{C}_1 and that the resulting completed PDAG of o' is \mathcal{C} .

The condition **iu**₁ clearly holds, since $x - y$ exists in \mathcal{C}_1 but not in \mathcal{C} . Lemma 15 implies that the graph obtained by deleting $x - y$ from \mathcal{C} is the completed PDAG \mathcal{C}_1 . Thus, the graph obtained by inserting $x - y$ into \mathcal{C}_1 is \mathcal{C} . This implies that InsertU $x - y$ is valid and the condition **iu**₂ holds.

Lemma 15 implies that the condition **iu**₃ also holds. \square

Proof of Lemma 8

PROOF. I will first show that there is no undirected edge $y - w$ that occurs in both \mathcal{C} and \mathcal{C}_1 . If $w - y$ occurs in \mathcal{C} , since x and y are not adjacent in

\mathcal{C} , $x \rightarrow w - y$ does not occur in \mathcal{C} . There are three possible configurations between x and w in \mathcal{C} : (1) x is not adjacent to w , (2) $w \rightarrow x$, and (3) $x - w$. If x is not adjacent to w in \mathcal{C} , inserting $x \rightarrow y$ will result in $y \rightarrow w$ in \mathcal{C}_1 . If $w \rightarrow x$ is in \mathcal{C} , inserting $x \rightarrow y$ will result in $w \rightarrow y$ in \mathcal{C}_1 . If $x - w$ in \mathcal{C} , there is an undirected path from y to x ; that is, the first condition for InsertD to be valid, according to Lemma 3, does not hold. Thus, we get that there is no undirected edge $y - w$ that occurs in both \mathcal{C} and \mathcal{C}_1 .

For any $w \in N_y$ in \mathcal{C}_1 , the edge between w and y is directed in \mathcal{C} ; that is, either $w \rightarrow y$ or $y \rightarrow w$ occurs in \mathcal{C} . If $y \rightarrow w$ is in \mathcal{C} , there are three possible configurations between x and w in \mathcal{C} : (1) x is not adjacent to w , (2) $w \rightarrow x$, and (3) $x \rightarrow w$. If x and w are not adjacent in \mathcal{C} , inserting $x \rightarrow y$ will result in $y \rightarrow w$ in \mathcal{C}_1 . If $w \rightarrow x$ occurs in \mathcal{C} , inserting $x \rightarrow y$ is not valid for \mathcal{C} since there would be a directed path from y to x . If $x \rightarrow w$ occurs in \mathcal{C} , w is common child of x and y , so from condition id_3 , $y \rightarrow w$ occurs in \mathcal{C}_1 and $w \notin N_y$ in \mathcal{C}_1 . Thus, we have that $w \rightarrow y$ must be in \mathcal{C} .

If there is another vertex $v \in N_y$ in \mathcal{C}_1 , $v \rightarrow y$ must also be in \mathcal{C} . If v and w are not adjacent, $v \rightarrow y \leftarrow w$ forms a v-structure both in \mathcal{C} and in \mathcal{C}_1 . $w \rightarrow y$ must occur in \mathcal{C}_1 and, consequently, $w \notin N_y$ in \mathcal{C}_1 yielding a contradiction. Thus, we know that any two vertices in N_y are adjacent in \mathcal{C} . N_y is therefore a clique in \mathcal{C}_1 and the operator DeleteD $x \rightarrow y$ is valid for \mathcal{C}_1 ; that is, the condition id_1 in Definition 9 holds.

Denote the modified PDAG of operator DeleteD $x \rightarrow y$ of \mathcal{C}_1 as \mathcal{P}' . We need to show that the corresponding completed PDAG of \mathcal{P}' is \mathcal{C} . Equivalently, we just need to show \mathcal{P}' and \mathcal{C} have the same skeleton and v-structures. Clearly, \mathcal{P}' and \mathcal{C} have the same skeleton. If there is a v-structure in \mathcal{C} , but not in \mathcal{C}_1 , it must be $x \rightarrow u \leftarrow y$, where u is a common child of x and y . From condition id_3 in Definition 9, $x \rightarrow u$ and $y \rightarrow u$ also occur in \mathcal{C}_1 , so, these v-structures also exist in \mathcal{P}' . This implies that all v-structures of \mathcal{C} are also in \mathcal{P}' . Moreover, the v-structures in \mathcal{C}_1 but not in \mathcal{C} must be $x \rightarrow y \leftarrow v$, where v is parent of y , and x and v are not adjacent in \mathcal{C}_1 . Clearly, after we delete $x \rightarrow y$ from \mathcal{C}_1 , these v-structures will not exist in \mathcal{P}' . This implies that all v-structures of \mathcal{P}' are in \mathcal{C} . So, \mathcal{P}' and \mathcal{C} have the same v-structures.

For any $v \rightarrow y$ in \mathcal{C}_1 , if $v - y$ is in \mathcal{C} , v must be parent of x . If x and v are not adjacent, inserting $x \rightarrow y$ to \mathcal{C} will result in $y \rightarrow v$ in \mathcal{C}_1 . Moreover, $x - v - y$ does not exist in \mathcal{C} since InsertD $x \rightarrow y$ is a valid operator, and $x \rightarrow v - y$ does not occur in \mathcal{C} . Thus, for any v that is a parent of y but not a parent of x , the directed edge $v \rightarrow y$ also occurs in the resulting completed PDAG \mathcal{C} . That is, the condition id_2 in Definition 9 holds.

□

Proof of Lemma 9

To prove this lemma, we first introduce Lemma 16 and Lemma 17. Let $L = (u_1, u_2, \dots, u_k)$ be a partially directed path from u_1 to u_k in a graph. A path $L_2 = (u^1, \dots, u^k)$ is a sub-path of L_1 if all vertices in L_1 are in L and have the same order as in L . We say that a partially directed path is shortest if it has no smaller sub-path.

LEMMA 16. *Let \mathcal{C} be a completed PDAG, and let L_1 be a partially directed path from y to x in \mathcal{C} . Then there exists a shortest sub-path of L_1 , denoted as $L_2 = y - u_1 - \dots - u_k \rightarrow \dots \rightarrow x$, in which there exists a k such that all edges occurring before u_k in the path are undirected, and all edges occurring after u_k are directed.*

PROOF. We just need to show that a directed edge must be followed by a directed edge in the shortest sub-path. If not, $u_i \rightarrow u_{i+1} - u_{i+2}$ occurs in L_2 . Because \mathcal{C} is a completed PDAG, u_i and u_{i+2} must be adjacent; otherwise $u_{i+1} \rightarrow u_{i+2}$ occurs in \mathcal{C} . If $u_i \rightarrow u_{i+2}$ occurs in \mathcal{C} , L_2 is not a shortest path. If $u_i \leftarrow u_{i+2}$ occurs in \mathcal{C} , $u_{i+1} \leftarrow u_{i+2}$ must be in \mathcal{C} . □

LEMMA 17. *If the graph \mathcal{P}_1 obtained by deleting $a \rightarrow b$ from a completed PDAG \mathcal{C} can be extended to a new completed PDAG, \mathcal{C}_1 , then we have that for any directed edge $x \rightarrow y$ in \mathcal{C} , if y is not b or a descendant of b , then $x \rightarrow y$ occurs in \mathcal{C}_1 .*

PROOF. Because $x \rightarrow y$ occurs in \mathcal{C} , so it is strongly protected in \mathcal{C} . If $x \rightarrow y$ does not occur in \mathcal{C}_1 , it is not strongly protected in \mathcal{C}_1 from Lemma 2. From the definition of strongly protected, we know that the four cases in Figure 1 in which $v \rightarrow u$ is strongly protected do not involve any descendant of u . Thus, if $x \rightarrow y$ is not compelled in \mathcal{C}_1 , there must exist a directed edge $w \rightarrow z$ between two non-descendants of y such that the edges between non-descendants of z are strongly protected and $w - z$ is no longer strongly protected in \mathcal{P}_1 . Because \mathcal{P}_1 is obtained by deleting $a \rightarrow b$, z is non-descendant of b , we have that $w \rightarrow z$ is strongly protected in \mathcal{P}_1 , yielding a contraction. □

We now give a proof of Lemma 9

PROOF. Since $\mathcal{C} \in \mathcal{S}_p^n$, we have $n_{\mathcal{C}_1} < n$. That is, the condition id_1 in Definition 9 holds for $\text{InsertD } x \rightarrow y$ of \mathcal{C}_1 .

For any undirected edge $w - y$ in \mathcal{C} , x must be parent of w , otherwise the edge between y and w is directed. Then deleting $x \rightarrow y$ from \mathcal{C} will result

in $w \rightarrow y$ in \mathcal{C}_1 . Thus, we have that all N_y in \mathcal{C} become parents of y in \mathcal{C}_1 . From the condition **dd**₂, the parents of y but not x in \mathcal{C} are also parents of y in \mathcal{C}_1 . If there is a partially directed path from y to x in \mathcal{C}_1 , then the vertex adjacent to y in this path must be a child of y or a vertex that is parent of y and x in \mathcal{C} . We will show that if the vertex is not a parent of y and x in \mathcal{C} , there exists a contradiction.

If there is a partially directed path from y to x in \mathcal{C}_1 , we can find a shortest partially directed path like $y - u_1 - \dots - u_k \rightarrow \dots \rightarrow x$ from Lemma 16, denoted as L_1 . Any directed edge, say $u_i \rightarrow u_{i+1}$, in L_1 does not become $u_i \leftarrow u_{i+1}$ in \mathcal{C} . If L_1 does not include undirected edges in \mathcal{C}_1 , we have that the vertices of L_1 form a partially directed cycle in \mathcal{C} . We just need to show that the vertices of the undirected path L_1 also form a partially directed path in \mathcal{C} .

Suppose $y \rightarrow u_1$ occurs in \mathcal{C} . If $u_1 - u_2$ is undirected in \mathcal{C} , then $y \rightarrow u_2$ must occur in \mathcal{C} , consequently, L_1 will not be shortest in \mathcal{C}_1 . If $u_2 \rightarrow u_1$ occurs in \mathcal{C} , there exists a v-structure $u_2 \rightarrow u_1 \leftarrow y$ in \mathcal{C}_1 otherwise u_2 and y are adjacent, and L_1 is not the shortest path in \mathcal{C}_1 . Thus, $u_1 \rightarrow u_2$ must occur in \mathcal{C} . In this manner, we get that all edges in $y - u_1 - \dots - u_k \rightarrow \dots \rightarrow x$ are directed in \mathcal{C} and are directed from $u_i \rightarrow u_{i+1}$. This implies that there exists a partially directed cycle in \mathcal{C} . So, u_1 must be a parent of y and x in \mathcal{C} . We have $u_1 \in \Omega_{xy}$ and every partially directed path of \mathcal{C}_1 from y to x contains at least one vertex in Ω_{xy} .

Since all vertices in Ω_{xy} in \mathcal{C}_1 are parents of x and y in \mathcal{C} , if there are two vertices, say $w_1, w_2 \in \Omega_{xy}$, that are not adjacent, the subgraph $w_1 \rightarrow y \leftarrow w_2$ could be a v-structure in \mathcal{C}_1 . So, all vertices in Ω_{xy} in \mathcal{C}_1 are adjacent and Ω_{xy} is a clique.

We have that the parents of y in \mathcal{C}_1 $((\Pi_y)_{\mathcal{C}_1})$ is in the union of the parents and neighbors of y in \mathcal{C} $((\Pi_y \cup N_y)_{\mathcal{C}_1})$. If there is at least one neighbor u of y in \mathcal{C} , u must be child of x in \mathcal{C} and parent of y in \mathcal{C}_1 , so parents of x and y are not the same. If there is no neighbor of y in \mathcal{C} , the parents of y in \mathcal{C}_1 is the same as in \mathcal{C} except those vertices that are parents of x that is $(\Pi_y - \Pi_x)_{\mathcal{C}_1} = (\Pi_y - \Pi_x)_{\mathcal{C}}$. At the same time, from Lemma 17, the parents of x in \mathcal{C}_1 are also the parents of x in \mathcal{C} . Thus, the parents of x and y are not the same in \mathcal{C}_1 . From Lemma 3, we have that InsertD $x \rightarrow y$ is valid for \mathcal{C}_1 and the condition **id**₂ holds.

Denote the modified PDAG of operator InsertD $x \rightarrow y$ of \mathcal{C}_1 as \mathcal{P}' . We need to show that the corresponding completed PDAG of \mathcal{P}' is \mathcal{C} . Equivalently, we just need to show that \mathcal{P}' and \mathcal{C} have the same skeleton and v-structures. Clearly, \mathcal{P}' and \mathcal{C} have the same skeleton. A v-structures that is in \mathcal{C} but not in \mathcal{C}_1 must have the form $x \rightarrow y \leftarrow u$, where u is parent of

y but not adjacent to x . From condition **dd**₂ in Definition 9, $u \rightarrow y$ also occurs in \mathcal{C}_1 , so, such a v-structure must also exist in \mathcal{P}' . This implies that all v-structures of \mathcal{C} are also in \mathcal{P}' . Moreover, the v-structures in \mathcal{C}_1 but not in \mathcal{C} must have the form $x \rightarrow v \leftarrow y$, where v is a common child of y and x in \mathcal{C}_1 . Clearly, after we insert $x \rightarrow y$ to \mathcal{C}_1 , this is no longer a v-structure in \mathcal{P}' implying that all v-structures of \mathcal{P}' are in \mathcal{C} . Thus, \mathcal{P}' and \mathcal{C} have the same v-structures.

Let the modified graph of DeleteD $x \rightarrow y$ from \mathcal{C} be \mathcal{P} ; we know that \mathcal{P} and \mathcal{C}_1 have the same v-structures. Thus, for any u that is a common child of x and y in \mathcal{C}_1 , $x \rightarrow u \leftarrow y$ is a v-structure in \mathcal{P} . This implies that $y \rightarrow u$ occurs in \mathcal{C} and the condition **id**₃ hold. \square

Proof of Lemma 10

PROOF. Since x, z and y are in the same chain component of \mathcal{C} , they have the same parent set in \mathcal{C} . The modified graph of o' has the same skeleton and v-structures as \mathcal{C}_1 because all compelled edges in \mathcal{C} remain compelled in \mathcal{C}_1 . We just need to prove that the operator o' is valid, equivalently, to prove that the conditions **rm**₁, **rm**₂ and **rm**₃ hold for \mathcal{C}_1 .

We now show that the condition **rm**₁, x and y have the same parents in \mathcal{C}_1 holds. Because x and y have the same parents in \mathcal{C} and all directed edges in \mathcal{C} occur in \mathcal{C}_1 , we just need to consider the neighbors of x or y . Let $w - y$ be any undirected edge in \mathcal{C} , we consider the edges between w and x or z .

1. If both $w - z$ and $x - w$ occur in \mathcal{C} , $w - y$ and $w - x$ must be undirected in \mathcal{C}_1 .
2. If $w - z$ occurs but $x - w$ does not occur in \mathcal{C} , $z \rightarrow w$ and $y \rightarrow w$ must be in \mathcal{C}_1 .
3. If $x - w$ occurs but $w - z$ does not occur in \mathcal{C} , there is an undirected cycle of length 4 without a chord in \mathcal{C} . Thus, this case will not occur.
4. If neither $w - z$ nor $x - w$ occur in \mathcal{C} , and there is no undirected path other than $w - y - z$ from w to z in \mathcal{C} , then $w - y$ occurs in \mathcal{C}_1 . If there exists another undirected path from w to z , there must exist an undirected path of length 2 like $w - u' - z$ in \mathcal{C} , and y is adjacent to u' . In this case, $y - w$ occurs in \mathcal{C}_1 when $x - u'$ occurs and $y \rightarrow w$ occurs when x and u' are not adjacent.

Thus, there are no neighbors of y in \mathcal{C} that become parents of y in \mathcal{C}_1 ; i.e., y has the same parents in both \mathcal{C}_1 and \mathcal{C} . Similarly, x has the same parents in both \mathcal{C}_1 and \mathcal{C} . we get x and y have the same parents in \mathcal{C}_1 , and the condition **rm**₁ holds.

All parents of x must also be parents of z in \mathcal{C}_1 since they are in the same chain component. For any $w \in N_{xy}$, $w - z$ also occurs in \mathcal{C} ; otherwise $x - z - y - w - x$ would form cycle of length 4 without a chord. We have $w \rightarrow z$ must be in \mathcal{C}_1 , otherwise a new v-structure will occur in \mathcal{C}_1 . Thus, we have $\Pi(x) \cup N_{xy} \subset \Pi(z)$ in \mathcal{C}_1 .

For any $w \in \Pi(z)$ in \mathcal{C}_1 , if $w \in \Pi(z)$ in \mathcal{C} , it must also be parent of x, y and z in \mathcal{C}_1 , so $w \in \Pi(x)$ in \mathcal{C}_1 . If $w - z$ is an undirected edge in \mathcal{C} , there exist undirected edges $w - x$ and $w - y$ in \mathcal{C} such that $w \rightarrow z$ is in \mathcal{C}_1 . Thus, $w \in N_{xy}$ in \mathcal{C}_1 . We have that $w \in \Pi(x) \cup N_{xy}$ and $\Pi(z) \subset \Pi(x) \cup N_{xy}$ in \mathcal{C}_1 . Thus, $\Pi(z) = \Pi(x) \cup N_{xy}$ in \mathcal{C}_1 and the condition **rm**₂ holds.

Any undirected path between x and y in \mathcal{C}_1 will also be an undirected path in \mathcal{C} , so, these paths contain at least one vertex in N_{xy} in \mathcal{C} . From the proof above, any vertex in N_{xy} in \mathcal{C} is also a vertex of N_{xy} in \mathcal{C}_1 . Thus, any undirected path between x and y contains a vertex in N_{xy} in \mathcal{C}_1 and the condition **rm**₃ holds. \square

Proof of Lemma 11

PROOF. From Lemma 5 and the condition **rm**₃, there exists a consistent extension of \mathcal{C} , denoted by \mathcal{D} , such that all neighbors of x in \mathcal{C} are children of x in \mathcal{D} , and all neighbors of y in \mathcal{C} are parents of x in \mathcal{D} . Changing $y \rightarrow z$ to $z \rightarrow y$ in \mathcal{D} , we obtain a new graph \mathcal{D}' . From the proof of Lemma 4, we can get that (1) \mathcal{D}' is a DAG, (2) \mathcal{D}' is a consistent extension of \mathcal{C}_1 . Thus, \mathcal{D} is a consistent extension of the PDAG that results from making the v-structure $x \rightarrow z \leftarrow y$ in \mathcal{C}_1 . Thus, we can get \mathcal{C} by applying MakeV $x \rightarrow z \leftarrow y$ to \mathcal{C}_1 . This implies that MakeV $x \rightarrow z \leftarrow y$ is a valid operator of \mathcal{O}_1 and satisfies the condition mv₁. \square

Proof of Theorem 6.

In order to prove this theorem, we first prove some results shown in Lemma 18, Lemma 19 and Lemma 20.

LEMMA 18. *For any completed PDAG \mathcal{C} containing at least one undirected edge, there exists an undirected edge $x - y$ for which N_{xy} is a clique.*

LEMMA 19. *For any completed PDAG \mathcal{C} , if $x \rightarrow y$ occurs in \mathcal{C} , then $\Pi_x \neq \Pi_y \setminus x$.*

A proof of Lemma 18 and Lemma 19 can be found in Chickering [3].

LEMMA 20. *For any completed PDAG \mathcal{C} containing no undirected edges and at least one directed edge, there exists at least one vertex x for which any parent of x has no parent.*

PROOF. The following procedure will find the vertex whose parent has no parent. Let $a \rightarrow b$ be a directed edge in \mathcal{C} , set $y = a$ and $x = b$.

1. If Π_y is not empty, choose any vertex u in Π_y , set $x = y$ and $y = u$. Repeat this step until we find a directed edge $y \rightarrow x$ for which Π_y is empty.
2. Since Π_y is empty, from Lemma 19, there exists at least one vertex other than y in Π_x . If there is a vertex $u \in \Pi_x$ and $u \neq y$ such that Π_u is not empty, choose a vertex in Π_u , denoted as v and set $y = v$ and $x = u$, and go to step 1.

Since \mathcal{C} is an acyclic graph with finite vertices, above procedure must end at the step in which the parents of x have no parents. \square

We now show a proof of Theorem 6.

PROOF. We need to show that for any two completed PDAGs $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{S}$, there exists a sequence of operators in \mathcal{O} such that \mathcal{C}_2 can be obtained by applying a sequence of operators to PDAGs, starting from \mathcal{C}_1 . Because \mathcal{O} is reversible, any operator in \mathcal{O} has a reversible operator, so we just need to show that any completed PDAG can be transferred to empty graph without edges. The procedure includes three basic steps.

1. Deleting all undirected edges.

From Lemma 18, for any completed PDAG containing at least one undirected edge, we can find an operator with type of DeleteU that satisfies the condition **du**₁ in Definition 9. We can delete an undirected edge with this operator and get a new completed PDAG whose skeleton is a subgraph of the skeleton of the initial completed PDAG. Repeating this procedure, we can get a completed PDAG, denoted as \mathcal{C}_i , which contains no undirected edges.

2. Deleting some directed edges.

From Lemma 20, we can find a vertex, denoted as x , whose parents have no parents in the completed PDAG \mathcal{C}_i . If Π_x contains more than two vertices, we can choose a vertex $u \in \Pi_x$. Because (1) N_x is empty in \mathcal{C}_i and (2) any other directed edge $v \rightarrow x$ forms a v-structure in \mathcal{C}_i , we have that $v \rightarrow x$ is also compelled in the completed PDAG obtained by deleting directed edge $u \rightarrow x$ from \mathcal{C}_i . We can delete $v \rightarrow x$ from \mathcal{C}_i and get a new completed PDAG whose skeleton is a subgraph of the skeleton of the initial one. Thus, the new

completed PDAG is in \mathcal{S} . Repeat this procedure for all other directed edges $v' \rightarrow x$ in which $v' \in \Pi_x$ until there are only two vertices in Π_x in the new completed PDAG, denoted as \mathcal{C}_j .

3. Remove a v-structure.

The conditions rm_1 , rm_2 and rm_3 hold for the v-structure $y \rightarrow x \leftarrow u$ in \mathcal{C}_j , so, we can remove $y \rightarrow x \leftarrow u$ from \mathcal{C}_j and get a new completed PDAG whose skeleton is a subgraph of the skeleton of the initial graph. Denote the resulting completed PDAG as \mathcal{C}_k , it may still contain some undirected edges.

By repeatedly applying the above the steps in sequence, we can finally obtain a graph without any edges. \square

3.2. *Proof of Theorem 2 introduced in Subsection 2.3.* There are three statements in Theorem 2; we prove them one by one below.

Proof of (i) of Theorem 2

(If)

Figure 1 shows the four cases that ensure that an edge is strongly protected. We first show that for any edge $x \rightarrow u$ (or $y \rightarrow u$), where u is a common child of x and y , if $x \rightarrow u$ is strongly protected in \mathcal{P}_{t+1} by configuration (a), (b), or (d) in Figure 1 (replace $v \rightarrow u$ by $x \rightarrow u$), it is also directed in e_{t+1} .

Case (1), (2) and (3) in Figure 12 show the sub-structures of \mathcal{P}_{t+1} in which $x \rightarrow u$ is protected by case (a), (b) and (d) in Figure 1 respectively, where \mathcal{P}_{t+1} is the modified graph obtained by inserting $x - y$ into e_t .

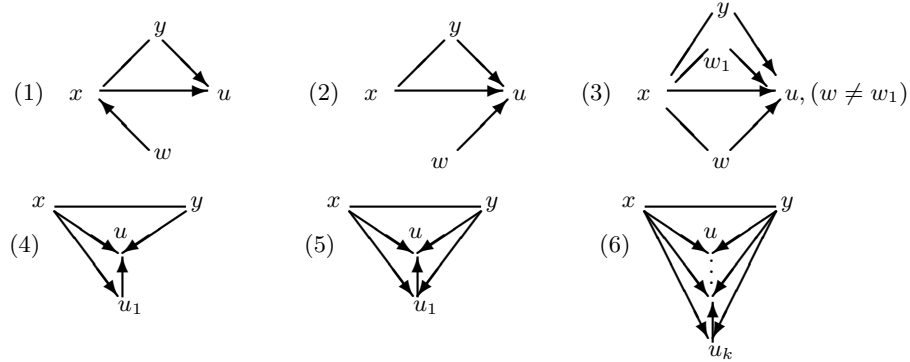


FIG 12. *strongly protected in \mathcal{P}_{t+1}*

If $x \rightarrow u$ is protected in \mathcal{P}_{t+1} like case (1) in Figure 12, $w \rightarrow x \rightarrow u$ occurs and w and u are not adjacent in \mathcal{P}_{t+1} . If $w \rightarrow x$ is undirected in e_{t+1} , from

Lemma 14, there exists a directed path from y to x . Any parent of x that is in this path must not be a parent of y ; otherwise, there exists a directed cycle from y to y in e_t . Hence we have that the parent sets of y and x are not equal. This is a contradiction of the condition $\Pi_x = \Pi_y$ in Lemma 3. We have that $w \rightarrow x$ and $x \rightarrow u$ occur in e_{t+1} .

If $x \rightarrow u$ is protected in \mathcal{P}_{t+1} by v-structure $x \rightarrow u \leftarrow w$, like case (2) in Figure 12, clearly, the v-structure also occurs in e_{t+1} , so $x \rightarrow u$ occurs in e_{t+1} .

If $x \rightarrow u$ is protected in \mathcal{P}_1 like case (3) in Figure 12, we have that the v-structure $w \rightarrow u \leftarrow w_1$ also occurs in e_{t+1} . If either $x \rightarrow u$ or $u \rightarrow x$ is in e_{t+1} , we have that $w_1 \rightarrow x$ and $w \rightarrow x$ are both in e_{t+1} and the v-structure $w_1 \rightarrow x \leftarrow w$ occurs. Hence we have that $x \rightarrow u$ occurs in e_{t+1} .

Now we show that if $x \rightarrow u$ is protected in \mathcal{P}_{t+1} like (c) in Figure 1, it is also protected in e_{t+1} . For any u_1 in $x \rightarrow u_1 \rightarrow u$, there are only two cases: u_1 and y are adjacent or nonadjacent.

When u_1 and y are not adjacent, like (4) in Figure 12, there is a v-structure $u_1 \rightarrow u \leftarrow y$ in \mathcal{P}_{t+1} . Then $u_1 \rightarrow u$ occurs in e_{t+1} . If $x \rightarrow u$ occurs in e_{t+1} , by Lemma 12, the edge between x and u_1 must be directed and oriented as $u_1 \rightarrow x$ in e_{t+1} . This is impossible, because there exists some extension of \mathcal{P}_{t+1} that has an edge oriented as $x \rightarrow u_1$. Thus, $x \rightarrow u$ occurs in e_{t+1} .

When u_1 and y are adjacent, we have that $u_1 \rightarrow y$ and $u_1 - y$ do not occur in \mathcal{P}_{t+1} since $P_x = P_y$ must hold in e_t for the validity of the operator InsertU $x - y$. Hence we have that $y \rightarrow u_1$ occurs in \mathcal{P}_{t+1} and $x \rightarrow u$ is strongly protected like case (5) in Figure 12. We consider two cases: $x \rightarrow u_1$ occurs or does not occur in e_{t+1} .

Assume $x \rightarrow u_1$ occurs in e_{t+1} . If $u_1 \rightarrow u$ occurs in e_{t+1} , clearly, $x \rightarrow u$ must occur in e_{t+1} because there is a partially directed path $x \rightarrow u_1 \rightarrow u$ in e_{t+1} . If $u_1 \rightarrow u$ is undirected in e_{t+1} , from Lemma 12, $x \rightarrow u$ must occur in e_{t+1} .

In case (5), we have that u_1 is also a common child of x and y , so, $x \rightarrow u_1$ will also be strongly protected in \mathcal{P}_{t+1} from the condition **iu**₃. Now, consider $x \rightarrow u_1$; if it is protected in \mathcal{P}_{t+1} like any of case (1), (2), (3), or (4), then, by our proof, $x \rightarrow u_1$ occurs in e_{t+1} . Thus, $x \rightarrow u$ must occur in e_{t+1} . If $x \rightarrow u_1$ is protected in \mathcal{P}_{t+1} like case (5), we can find another vertex u_2 that is a common child of x and y like case (6). From the proof above, we know if $x \rightarrow u_2$ occurs in e_{t+1} , $x \rightarrow u_1$ and $x \rightarrow u$ also occur in e_{t+1} . Since the graph has finite vertices, we can find a common child of x and y , say u_k , such that $x \rightarrow u_k$ is protected in \mathcal{P}_{t+1} like one of cases (1), (2), (3) or (4). Thus, $x \rightarrow u_k$ occurs in e_{t+1} , implying that $x \rightarrow u_{k-1}$ occurs in \mathcal{P}_{t+1} , so, finally, $x \rightarrow u$ occurs in \mathcal{P}_{t+1} .

(**Only if**) From Lemma 15, we have that the modified graph \mathcal{P}_{t+1} is also the resulting completed PDAG e_{t+1} . Hence, all directed edges in e_{t+1} are strongly protected in \mathcal{P}_{t+1} , so the Algorithm 1.1.1 will return True.

□

Proof of (ii) of Theorem 2

To prove (ii) of Theorem 2, we need following lemma.

LEMMA 21. *Let e_t be a completed PDAG, \mathcal{P}_{t+1} be the PDAG obtained in Algorithm 1.1.2 with input of a valid operator $\text{InsertD } x \rightarrow y$, and e_{t+1} be the resulting completed PDAG extended from \mathcal{P}_{t+1} . We have:*

1. *If u is not a common child of x and y , then all directed edges $y \rightarrow u$ in \mathcal{P}_{t+1} are also in e_{t+1} .*
2. *All directed edges $v \rightarrow y$ in \mathcal{P}_{t+1} are also in e_{t+1} ;*

PROOF. (1)

If u is not a common child of x and y , and $y \rightarrow u$ occurs in \mathcal{P}_{t+1} , we have that there is a structure like $x \rightarrow y \rightarrow u$ in \mathcal{P}_{t+1} . Because $x \rightarrow y$ occurs in e_{t+1} , $y \rightarrow u$ must be in e_{t+1} too.

(2)

From Algorithm 1.1.2, all directed edges $v \rightarrow y$ are strongly protected in \mathcal{P}_{t+1} . When v is not adjacent to x in e_{t+1} , $v \rightarrow y \leftarrow x$ is a v-structure, so $v \rightarrow y$ occurs in e_{t+1} . When v is adjacent to x , we show below that if $v \rightarrow y$ is strongly protected like one of four cases in Figure 13, it is also strongly protected in e_{t+1} .

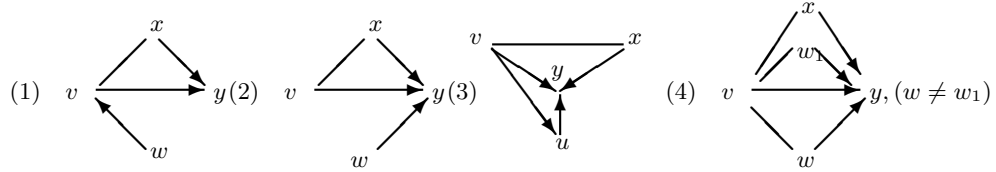


FIG 13. *strongly protected in \mathcal{P}_{t+1} .*

In case (1) of Figure 13, because there is no path from y to v , we have that $w \rightarrow v$ occurs in e_{t+1} from Lemma 14. Hence we have that $v \rightarrow y$ occurs in e_{t+1} .

In case (2), there is a v-structure $w \rightarrow y \leftarrow v$ in \mathcal{P}_{t+1} . So, $v \rightarrow y$ occurs in e_{t+1} .

In case (3), because there is no path from y to u , we have that $v \rightarrow u$ occurs in e_{t+1} according to Lemma 14. If $u \rightarrow y$ occurs in e_{t+1} , $v \rightarrow y$ occurs

in e_{t+1} . If $u \rightarrow y$ become $u - y$ in e_{t+1} , $v \rightarrow y$ must also be in e_{t+1} from Lemma 12.

From the proof of (i) of Theorem 2, we also have that $v \rightarrow y$ must be in e_{t+1} when case (4) occurs in \mathcal{P}_{t+1} .

Notice that the above proof also holds when we replace $x - v$ by a directed edge or add an edge between x and w (or u). Hence we have that $v \rightarrow y$ in \mathcal{P}_{t+1} also occurs in e_{t+1} . \square

We now give a proof for (ii) of Theorem 2.

(If)

We need to consider four cases in Figure 1 in which $y \rightarrow u$ is strongly protected in \mathcal{P}_{t+1} . Similar to the proof of (i) of Theorem 2, we first prove that the theorem holds in the first three cases in Figure 1, which correspond to the cases (1)', (2)' and (3)' shown in Figure 14. Notice that the following proof holds for any configuration of the edge between x and w .

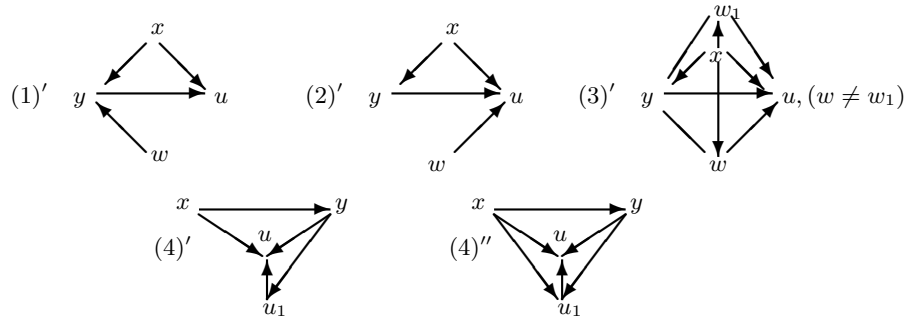


FIG 14. Five cases in which $x \rightarrow u$ or $y \rightarrow u$ is strongly protected.

Consider the case (1)' in Figure 14. From Lemma 21, $w \rightarrow y$ occurs in e_{t+1} . We have that $x \rightarrow u$ must occur in e_{t+1} .

Because there is a v-structure $w \rightarrow u \leftarrow y$ in case (2)', we have that $w \rightarrow u \leftarrow y$ also occurs in e_{t+1} .

After implementing Algorithm 1.1.2, if case (3)' occurs in \mathcal{P}_{t+1} , we have that $y - w$ is not strongly protected in \mathcal{P}_{t+1} and the edge between y and w have opposite directions in different consistent extensions of \mathcal{P}_{t+1} . Hence $y - w$ occurs in e_{t+1} . Similarly, $y - w_1$ also occurs in e_{t+1} . Moreover, the v-structure $w \rightarrow u \leftarrow w_1$ occurs in e_{t+1} . We have that $y \rightarrow u$ is strongly protected and occurs in e_{t+1} .

We now just need to show that a directed edge $y \rightarrow u$ that is strongly protected in \mathcal{P}_{t+1} like case (4)' (x and u_1 are nonadjacent) or (4)'' (x and

u_1 are adjacent) in Figure 14 is also directed in e_{t+1} .

In case (4)', from delete Lemma 21, $y \rightarrow u_1$ occurs in e_{t+1} . Moreover, $x \rightarrow u \leftarrow u_1$ is a v-structure, so $u_1 \rightarrow u$ also occurs in \mathcal{C}_1 . So we have $y \rightarrow u$ must occur in e_{t+1} .

In case (4)", we have that u_1 is also a common child of x and y ; hence, $y \rightarrow u_1$ will also be strongly protected in \mathcal{P}_{t+1} from the condition of this Theorem. Consider $y \rightarrow u_1$; if it is protected in \mathcal{P}_{t+1} like at least one case other than (4)", from our proof, $y \rightarrow u_1$ is also compelled in e_{t+1} , so $y \rightarrow u$ must be compelled in e_{t+1} . If $y \rightarrow u_1$ is protected in \mathcal{P}_{t+1} like case (4)", we can find another vertex u_2 that is a common child of y and x ; from the proof above, we know if $y \rightarrow u_2$ is directed in e_{t+1} , $y \rightarrow u_1$ and $y \rightarrow u$ are directed too. Since the graph has finite vertices, we can find a common child of x and y , say u_k , such that u_k is protected in \mathcal{P}_{t+1} like at least one case other than (4)". It is compelled in e_{t+1} , so we can get $y \rightarrow u_{k-1}$ is compelled in \mathcal{P}_{t+1} , so, finally, $y \rightarrow u$ is also compelled in \mathcal{P}_{t+1} . We have that $y \rightarrow u$ must occur in e_{t+1} and **id**₃ holds.

(Only if) Let u be a common child of x and y in e_t . If condition **id**₃ holds for a valid operator InsertD $x \rightarrow y$, we have that $y \rightarrow u$ in e_t occurs in e_{t+1} and is strongly protected in e_{t+1} . We need to show that $y \rightarrow u$ must be strongly protected in \mathcal{P}_{t+1} , obtained in Algorithm 1.1.2. From the proof of this statement above, we know we just need to consider the five configurations in which $y \rightarrow u$ is strongly protected in e_{t+1} in Figure 14.

We know that v-structures in e_{t+1} occur in \mathcal{P}_{t+1} therefore, the v-structure in the cases (2)', (3)' and (4)' in e_{t+1} must occur in \mathcal{P}_{t+1} too.

For case (2)', $y \rightarrow u$ is also strongly protected in \mathcal{P}_{t+1} , since the v-structure $y \rightarrow u \leftarrow w$ occurs in \mathcal{P}_{t+1} .

For case (3)', we have that (1) the v-structure $w_1 \rightarrow u \leftarrow w$ occurs in \mathcal{P}_{t+1} ; (2) e_{t+1} and \mathcal{P}_{t+1} have the same set of v-structures. Hence the v-structure $w_1 \rightarrow y \leftarrow w$ does not occur in \mathcal{P}_{t+1} . We have that $y \rightarrow u$ is also strongly protected in \mathcal{P}_{t+1} for any configuration of edges between w_1 , y and w .

For case (4)', from Algorithm 1.1.2, $y \rightarrow u_1$ occurs in \mathcal{P}_{t+1} . Hence $y \rightarrow u$ is strongly protected in \mathcal{P}_{t+1} .

Because the valid operator "Insert $x \rightarrow y$ " satisfies condition **id**₃, from Lemma 8, we have that the operator "Delete $x \rightarrow y$ ", when applied to e_{t+1} , results in e_t . From the condition **dd**₂, any directed edge $v \rightarrow y$ in e_{t+1} also occurs in e_t . For case (1)', we have that $v \rightarrow y \rightarrow u$ is strongly protected in \mathcal{P}_{t+1} .

Consider the case (4)", we have that v-structures $x \rightarrow u \rightarrow y$ and $x \rightarrow u_1 \rightarrow y$ occur in e_t since e_t is the resulting completed PDAG of the operator

“Delete $x \rightarrow y$ ” from e_{t+1} . According to Algorithm 1.1.2, $x \rightarrow y$, $x \rightarrow u \rightarrow y$ and $x \rightarrow u_1 \rightarrow y$ occur in \mathcal{P}_{t+1} . We have that $u \rightarrow u_1$ does not occur in e_t , otherwise $u \rightarrow u_1$ occurs in at least one consistent extension of \mathcal{P}_{t+1} and consequently $u_1 \rightarrow u$ does not occur in e_{t+1} . To prove that $y \rightarrow u$ is strongly protected in e_{t+1} , we need to show that $u_1 \rightarrow u$ occurs in e_t . Equivalently, we show $u_1 - u$ does not occur in e_t . If $u_1 - u$ occurs in a chain component denoted by τ in e_t , we have that neither x nor y are in τ . The undirected edges adjacent to x or y are in chain components different to τ . Hence **id**₃ holds for the operator “Insert $x \rightarrow y$ ”, and all parents of τ occur in e_{t+1} too. We have that $u_1 - u$ occurs in e_{t+1} too. It’s a contradiction that $u_1 \rightarrow y$ occurs in e_{t+1} . \square

Proof of (iii) of Theorem 2

(If)

Since Algorithm 1.1.3 returns True, all directed edges like $v \rightarrow y$ are strongly protected in \mathcal{P}_{t+1} . Consider the four configurations in which $v \rightarrow y$ is strongly protected in \mathcal{P}_{t+1} in Figure 15. Notice that \mathcal{P}_{t+1} is obtained by deleting $x \rightarrow y$ from completed PDAG \mathcal{C} , by Lemma 17, all directed edges with no vertices being descendants of y (excluding y) in \mathcal{P}_{t+1} will occur in e_t .

Hence, we have the edges $w \rightarrow v$ in case (1), and $v \rightarrow w$ in case (3) will remain in e_{t+1} . We have $v \rightarrow y$ in case (1) and case (3) must occur in e_{t+1} . Because v-structures in case (2) and case (4) will also remain in e_{t+1} , $v \rightarrow y$ in case (2) and case (4) must occur in e_{t+1} too.

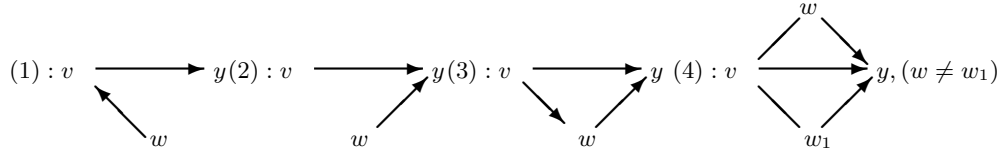


FIG 15. Four configurations of $v \rightarrow y$ being strongly protected.

(Only if) If condition **dd**₂ holds for a valid operator Deleted $x \rightarrow y$, all edges like $v \rightarrow y$ ($v \neq x$) in e_t will occur in e_{t+1} . $v \rightarrow y$ must be strongly protected in e_{t+1} . Consider the four configurations in which $v \rightarrow y$ is strongly protected in e_{t+1} as Figure 15. We know that v-structures in e_{t+1} must occur in e_t ; consequently, all directed edges in e_{t+1} must occur in e_t ; they also occur in \mathcal{P}_{t+1} . From Lemma 17, $w - v - w_1$ in case (4) in Figure 15 must be in \mathcal{P}_{t+1} , so an edge $v \rightarrow y$ that is strongly protected in e_{t+1} is also strongly protected in \mathcal{P}_{t+1} .

□

References.

- [1] S.A. Andersson, D. Madigan, and M.D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505–541, 1997.
- [2] D.M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *UAI95*, pages 87–98. Citeseer, 1995.
- [3] D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *The Journal of Machine Learning Research*, 2:445–498, 2002.
- [4] D. Dor and M. Tarsi. A simple algorithm to construct a consistent extension of a partially oriented graph. *Technical Report R-185, Cognitive Systems Laboratory, UCLA*, 1992.
- [5] Y. He, J. Jia, and B. Yu. Reversible mcmc on markov equivalence classes of sparse directed acyclic graphs. *arXiv preprint arXiv:1209.5860*, 2012.
- [6] S.L. Lauritzen and T.S. Richardson. Chain graph models and their causal interpretations. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64(3):321–361, 2002.

SCHOOL OF MATHEMATICAL SCIENCES, LMAM, LMEQF
 PEKING UNIVERSITY
 BEIJING 100871, CHINA
 E-MAIL: heyb@math.pku.edu.cn
 jzjia@math.pku.edu.cn

DEPARTMENT OF STATISTICS
 UC, BERKELEY, CA 94720
 E-MAIL: binyu@stat.berkeley.edu